Distributed Systems in Action

Mohamed Sweelam

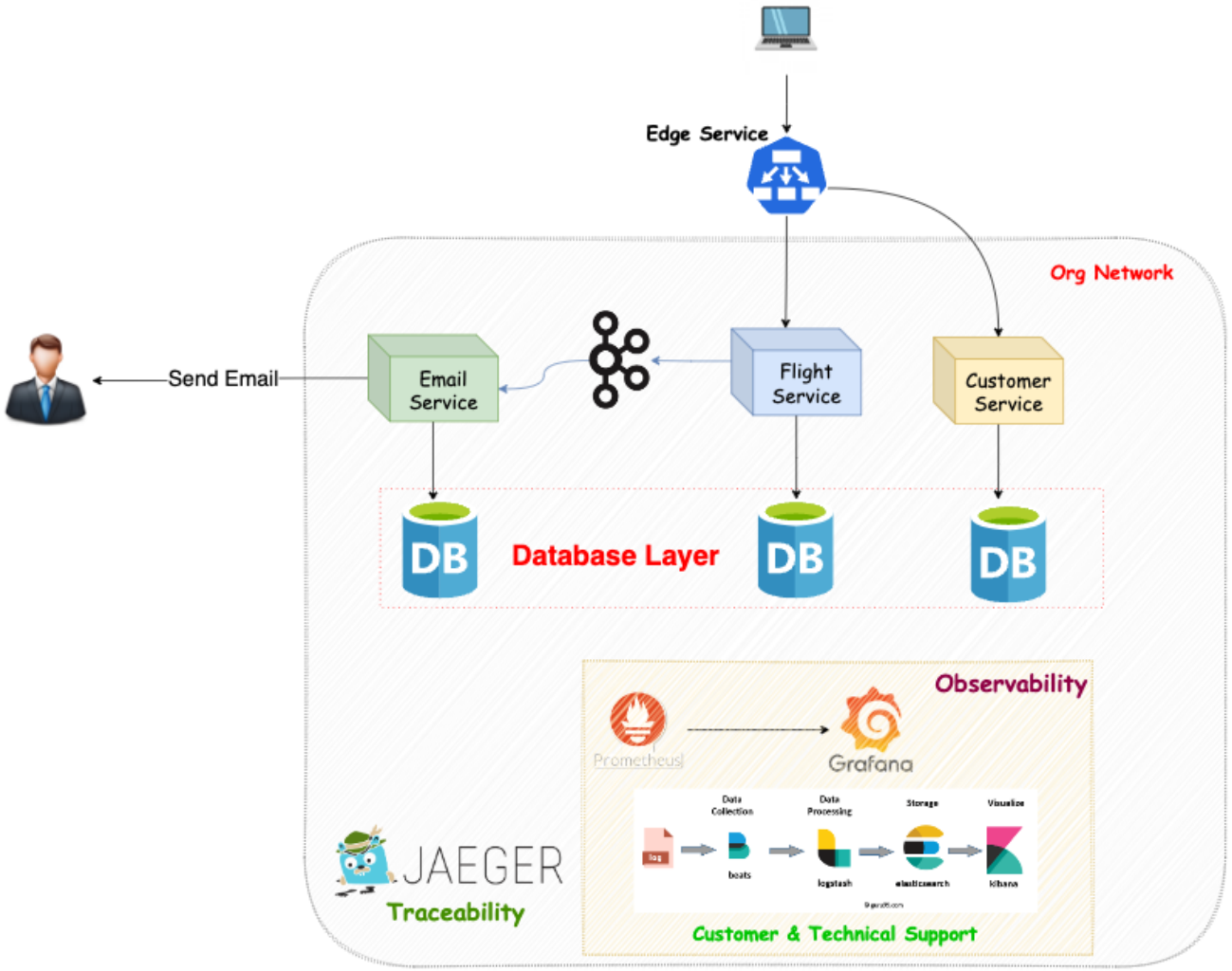
Software Engineer

Outline

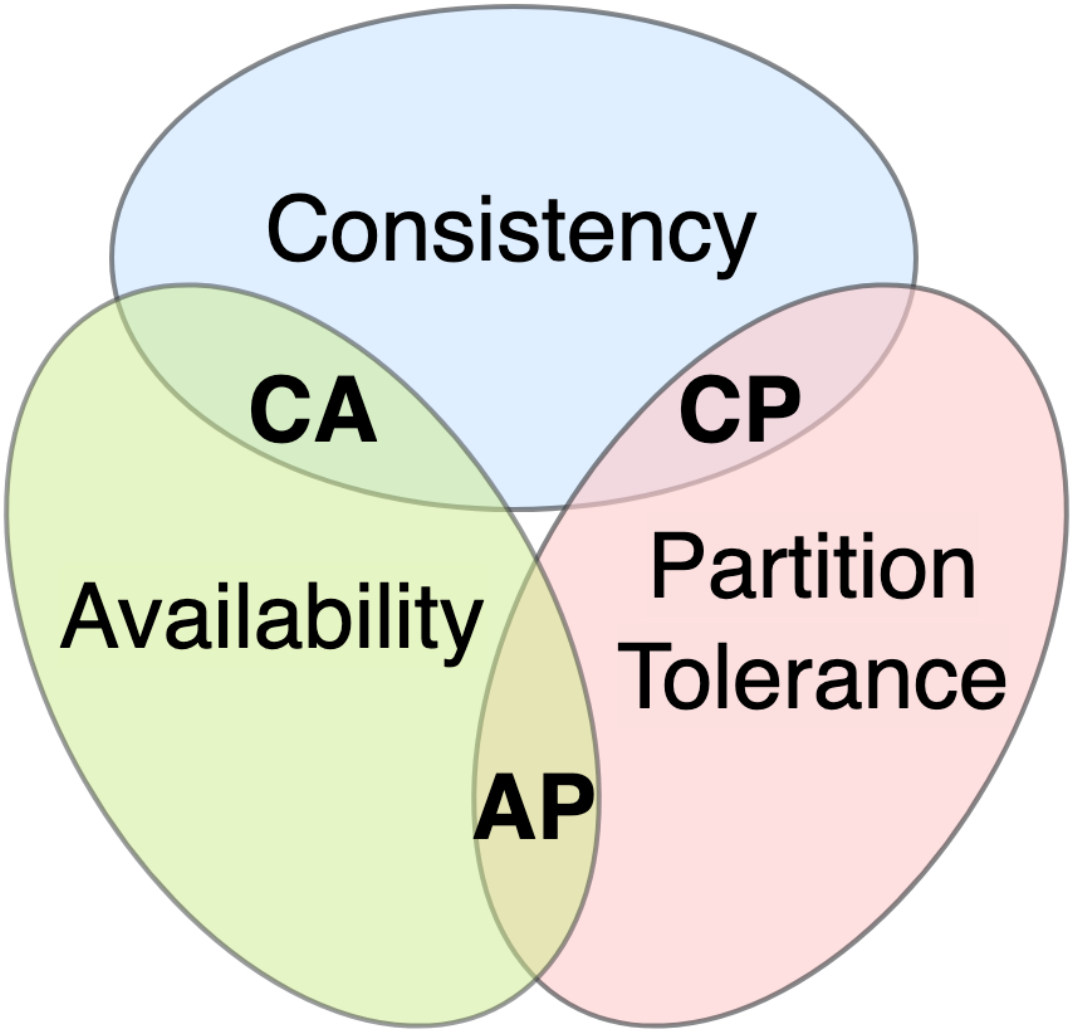
Introduction to Distributed Systems

- Definition and characteristics of distributed systems
- Importance and benefits of distributed systems
- Challenges and trade-offs in designing distributed systems

Use Case: Flight System

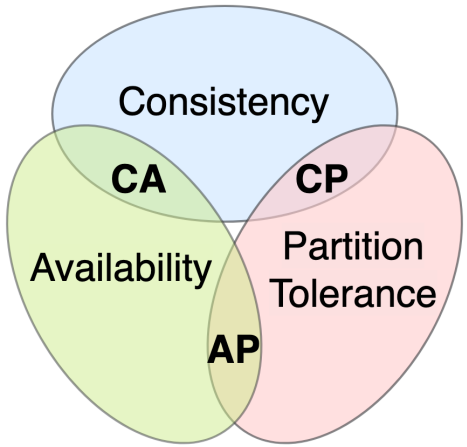


CAP Theorem



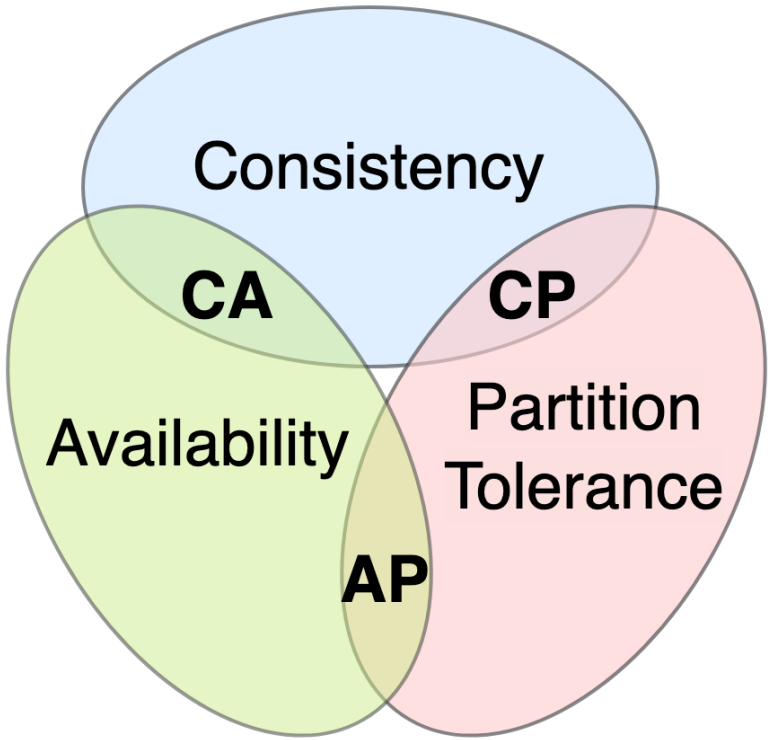
Distributed Databases

- **Consistency** every read operation gives a result of the most recent write.
- **Availability** every read operation gives a non-error response, but, data might be staled.
- **Partition Tolerance** system operates normally despite network failure.

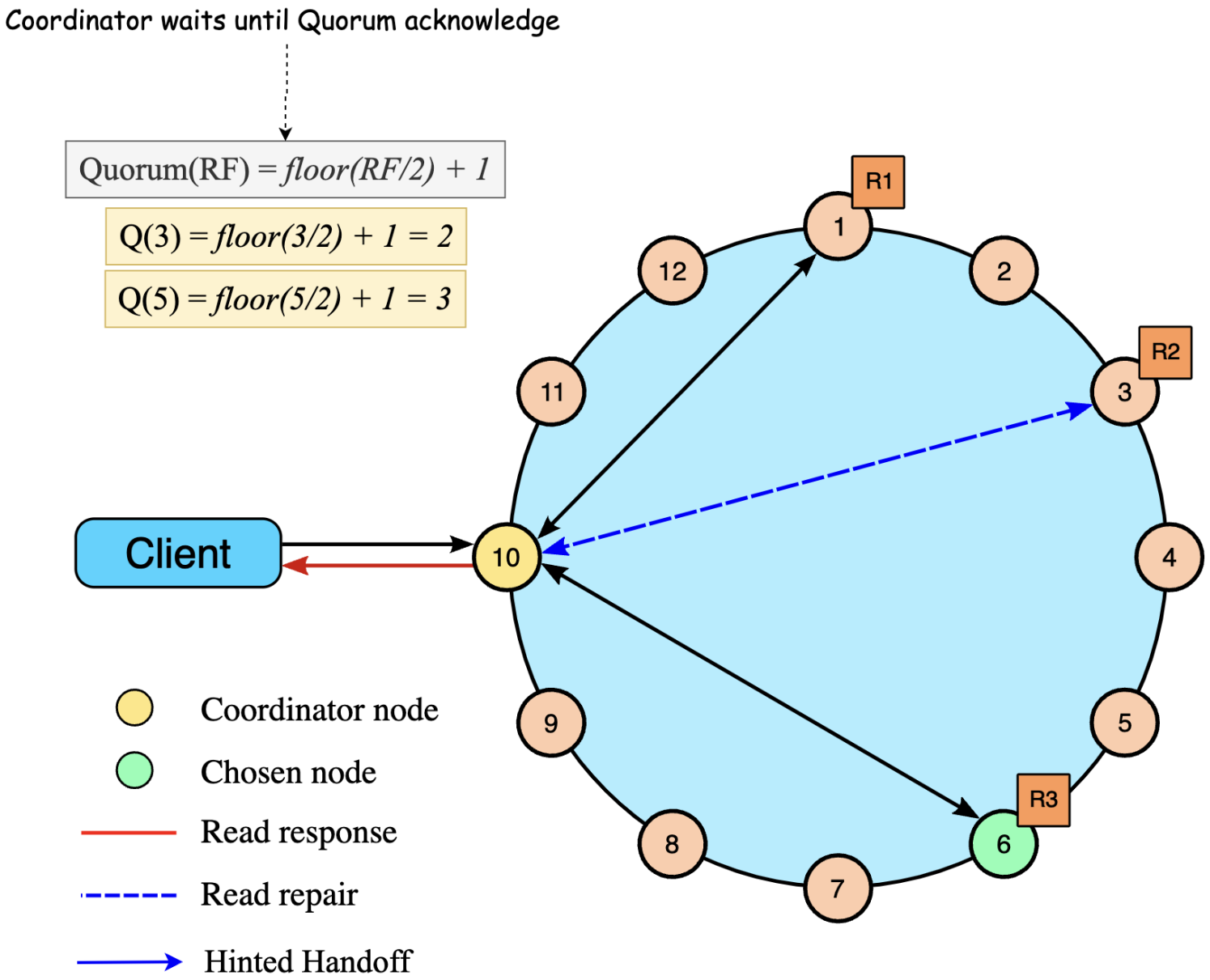


Distributed Databases

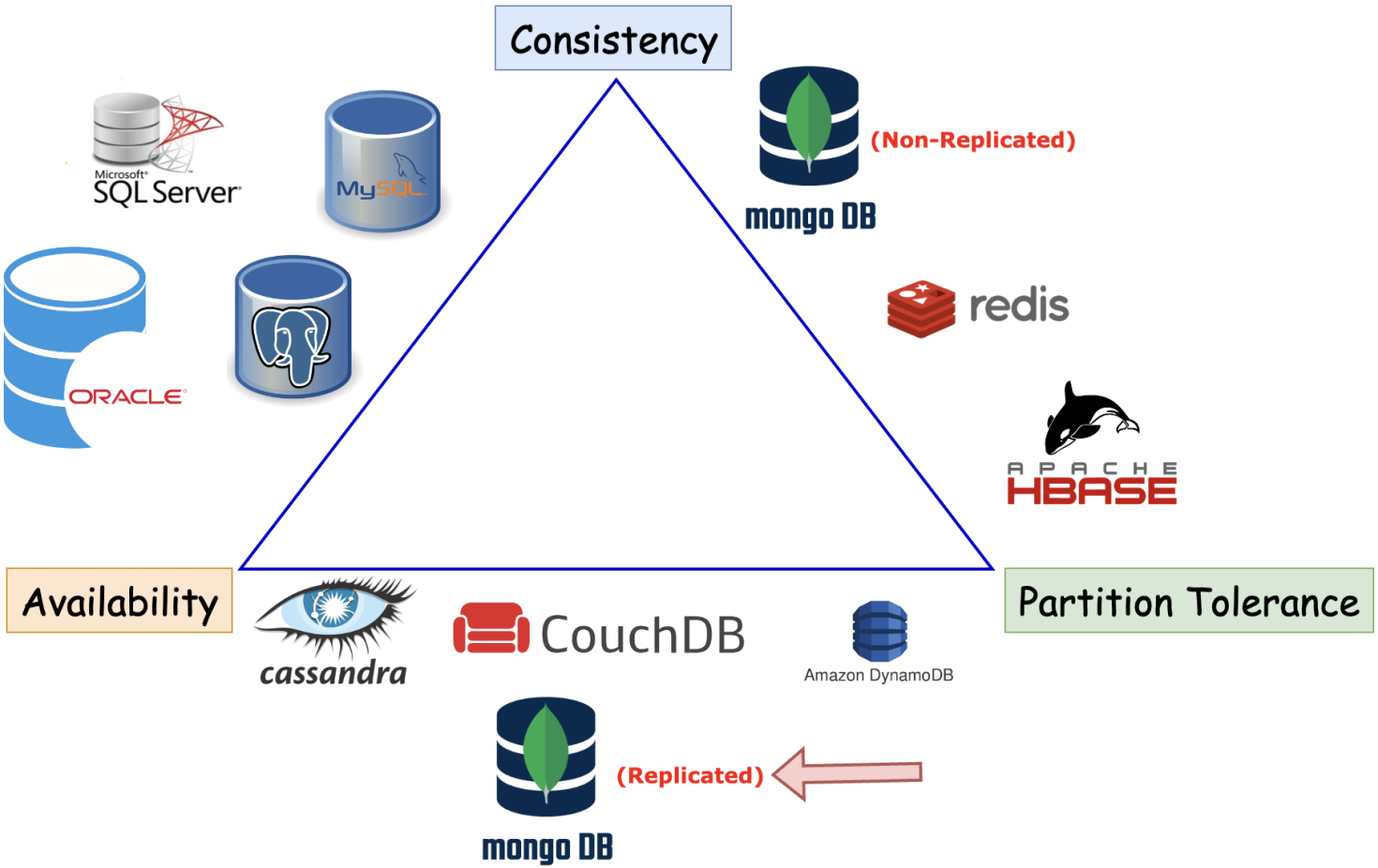
In the presence of network partitioning, system designers must choose between data consistency and availablity.



Distributed Databases

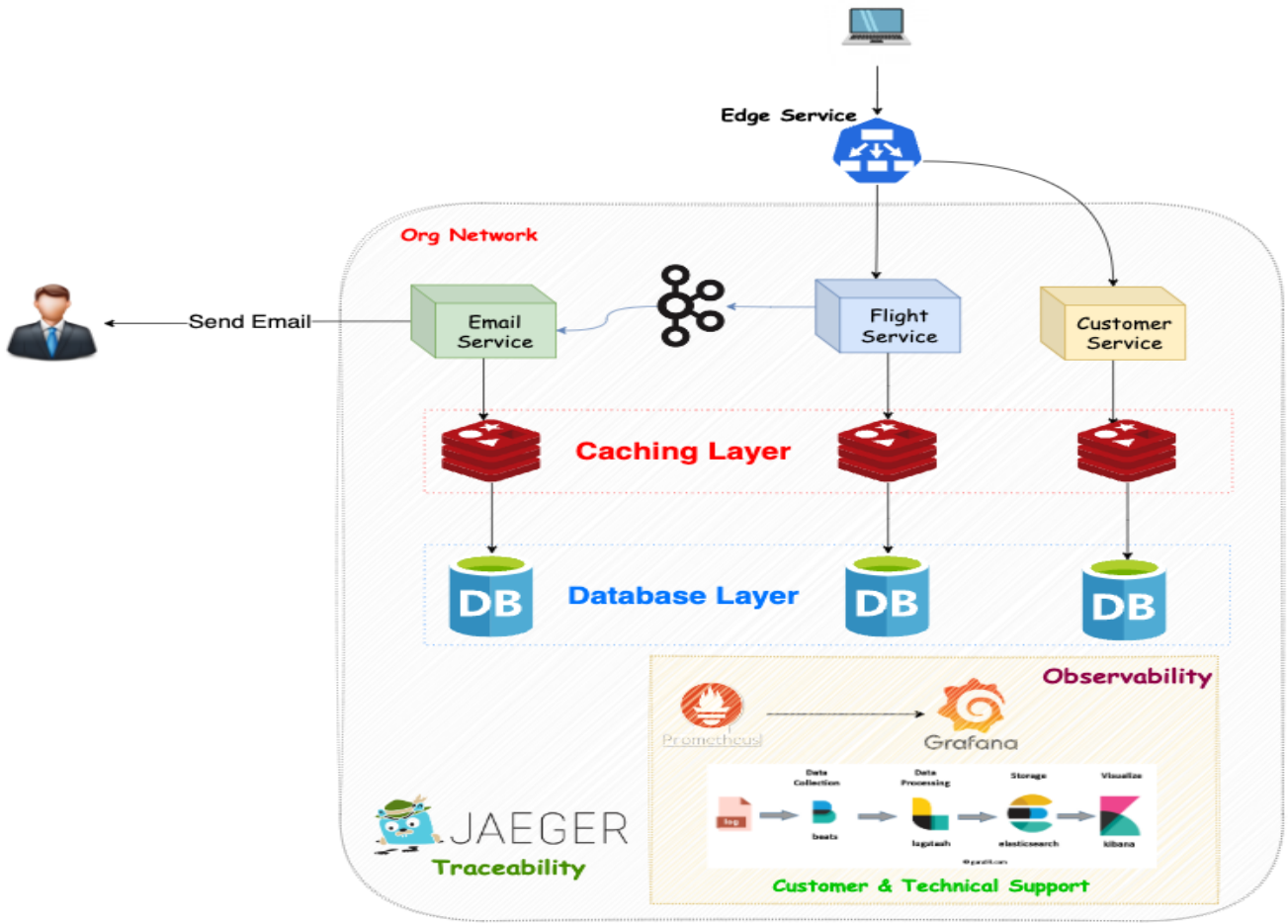


Distributed Databases



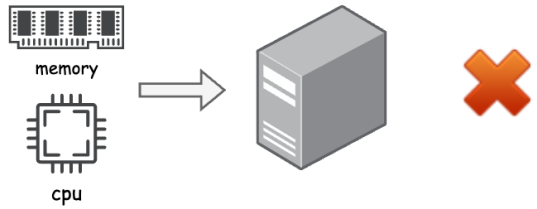
High Availability and Scalability

- Load Balancer
- API Gateway
- Rate Limiter



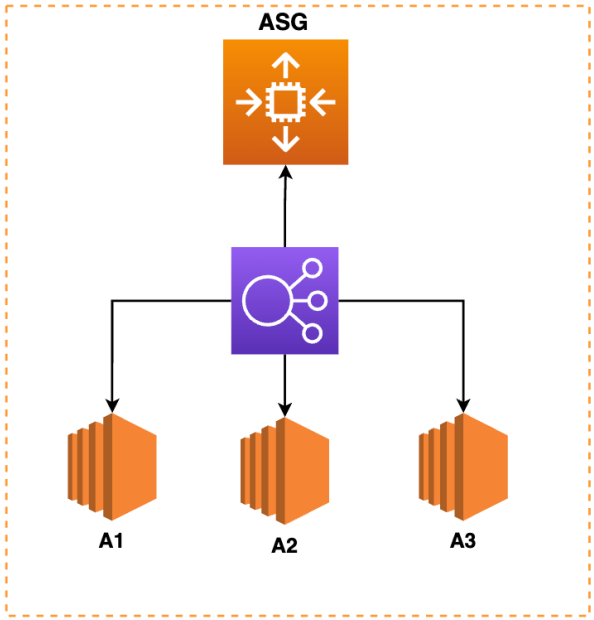
Horizontal Scaling vs Vertical Scaling

Vertical Scaling



Horizontal Scaling

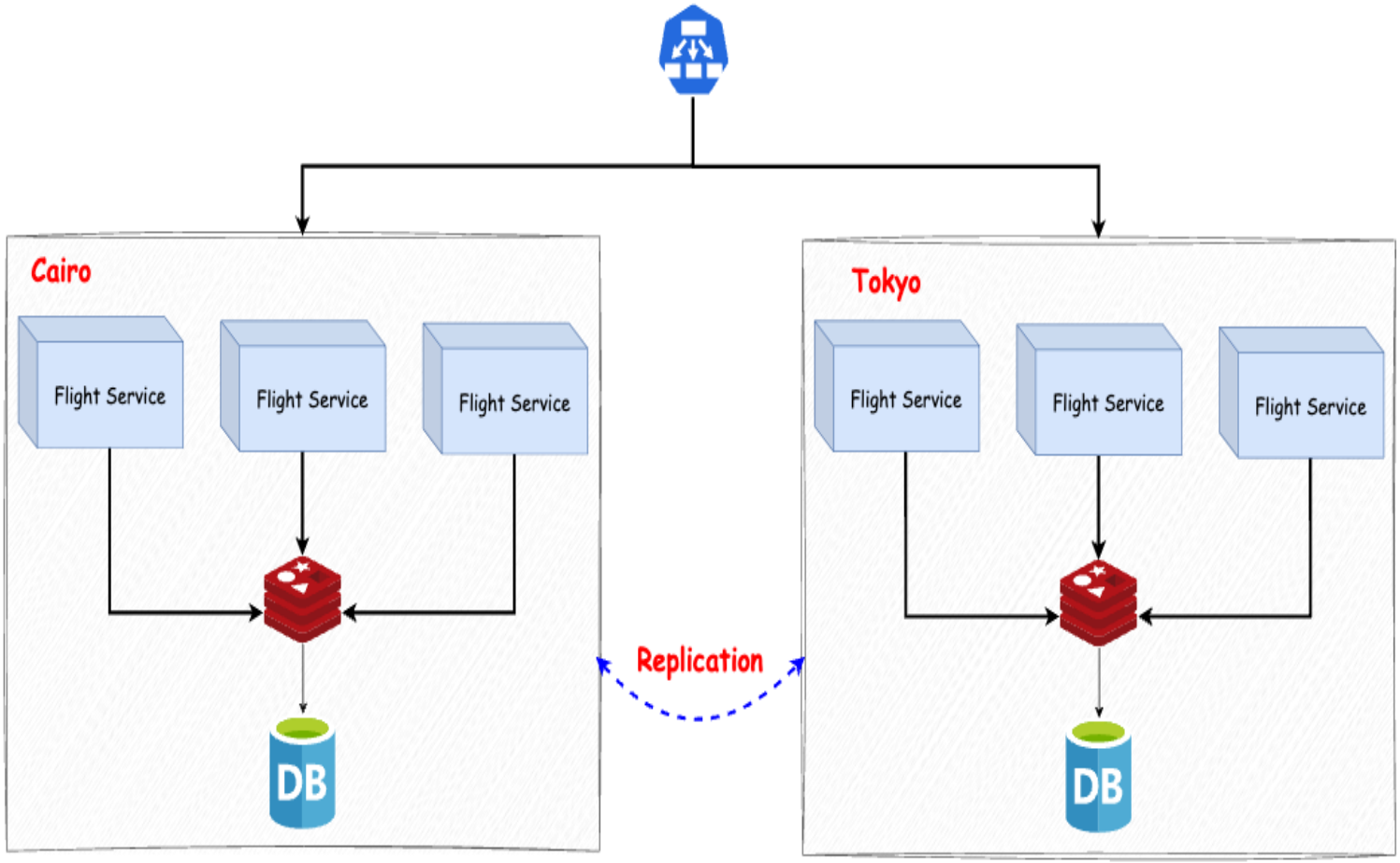

```yaml
spec:
  containers:
  - name: app
    image: images.my-company.example/app:v4
    resources:
      requests:
        memory: "64Mi"
        cpu: "250m"
      limits:
        memory: "128Mi"
        cpu: "500m"
```



Resiliency

```
spring:
  cloud:
    default-filters:
      - name: Retry
        args:
          retry: 5
          methods: GET
          backoff:
            firstBackOff: 10ms
            maxBackOff: 50ms
            factor: 2
            basedOnPreviousValue: false
```

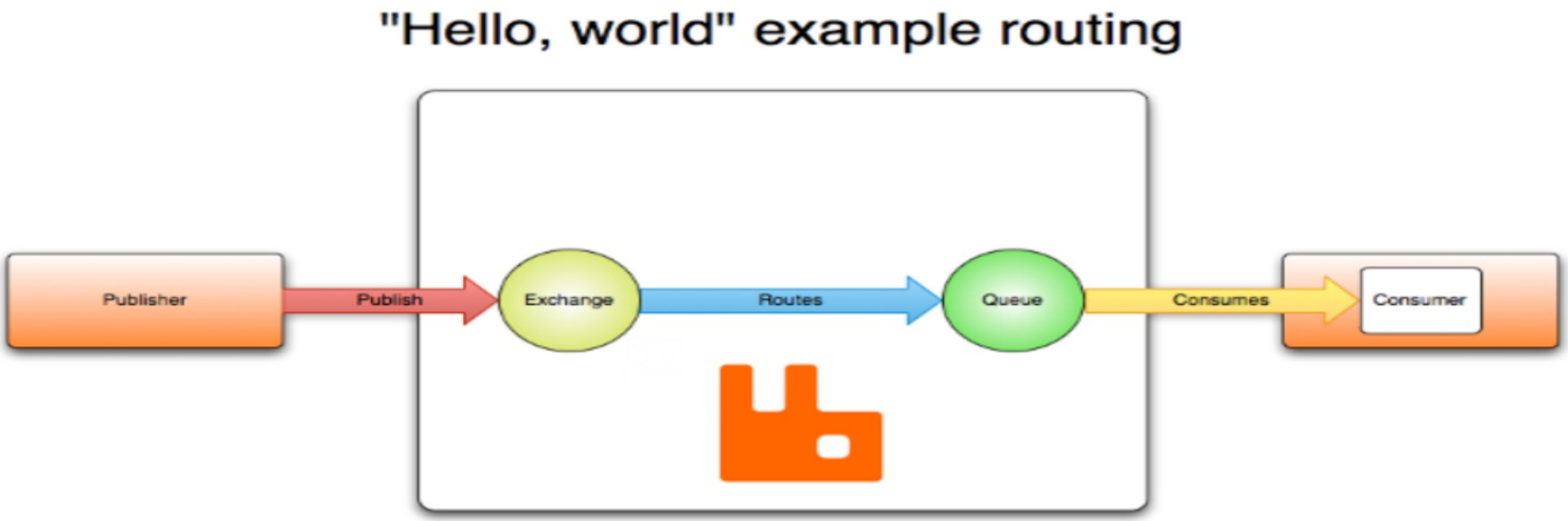
Data Replication

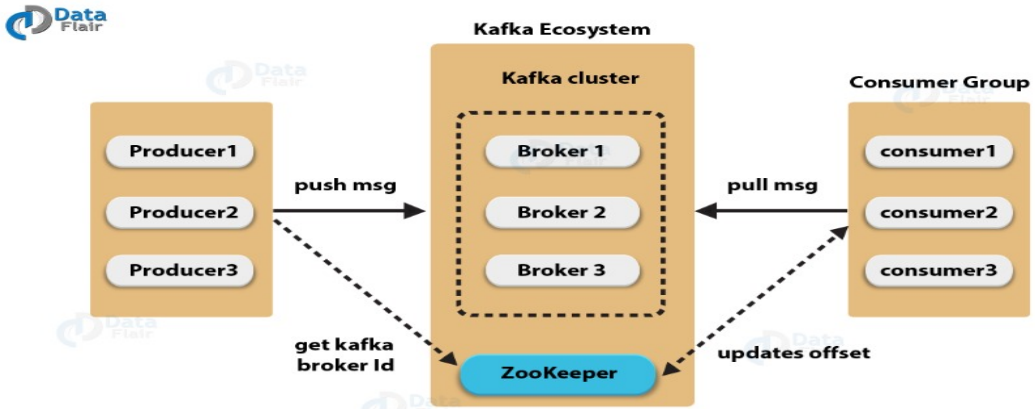


Distributed Messaging Systems

- Introduction to distributed messaging systems
- Overview of Kafka and RabbitMQ

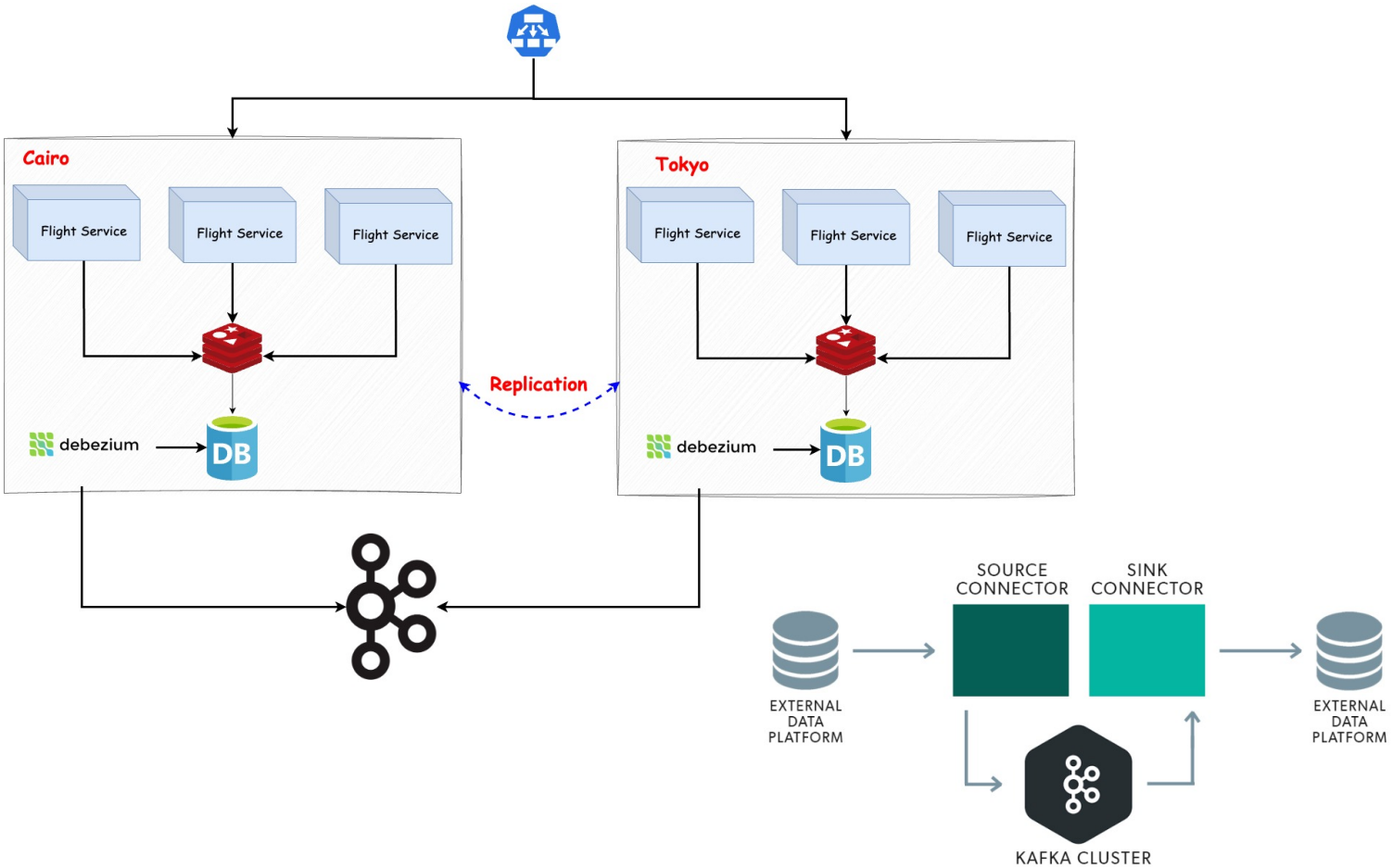
RabbitMQ vs Kafka





Data Replication: CDC

In databases, **change data capture (CDC)** is a set of *software design patterns* used to determine and track the data that has changed (the "deltas") so that action can be taken using the changed data.



Data Replication: CDC


```
{
  "schema": {},
  "payload": {
    "source": {
      "version": "2.2.1.Final",
      "connector": "mysql",
      "name": "mysql",
      "db": "inventory",
      "sequence": null,
      "table": "customers"
    },
    "databaseName": "inventory",
    "schemaName": null,
    "ddl": "ALTER TABLE customers ADD middle_name varchar(255) AFTER first_name",
    "tableChanges": [
      {
        "type": "ALTER",
        "id": "\"inventory\".\"customers\"",
        "table": {
          "defaultCharsetName": "utf8mb4",
          "primaryKeyColumnNames": [
            "id"
          ],
          "columns": [
            {
              "name": "id",
              "jdbcType": 4,
              "nativeType": null,
              "typeName": "INT",
              "typeExpression": "INT",
              "charsetName": null,
              "length": null,
              "scale": null,
              "position": 1,
              "optional": false,
              "autoIncremented": true,
              "generated": true
            }
          ]
        }
      }
    ]
  }
}
```

Distributed Messaging Systems

- Introduction to distributed messaging systems
- Overview of Kafka and RabbitMQ
- Comparison between Kafka and RabbitMQ:

Kafka	**RabbitMQ**
High-throughput, fault-tolerant distributed streaming platform.	Robust and flexible messaging broker.
Emphasizes real-time event streaming and data pipeline use cases.	Implements Advanced Message Queuing Protocol (AMQP).
Provides strong durability and replication guarantees.	Focuses on message queuing and asynchronous communication.
Scales horizontally to handle large-scale data streams.	Provides various messaging patterns (e.g., publish-subscribe, point-to-point).
Supports complex event processing with built-in stream processing.	Offers pluggable message durability, routing, and acknowledgement mechanisms.

Distributed Key-Value Stores

A type of NoSQL DBs, the idea behind having **it** is having an extremely fast mechanism for retrieving the data.

- Introduction to distributed key-value stores (e.g., Redis, Memcached)
- Data partitioning and replication techniques
- Consistency and availability trade-offs

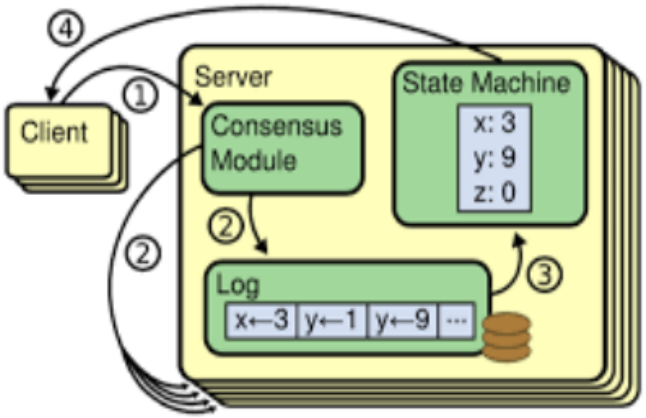
Distributed Key-Value Stores: Memcached vs Redis

Memcached	**Redis**
In-memory key-value store primarily designed for caching.	In-memory key-value store with additional data structures and functionality.
Supports simple key-value operations like GET, SET, DELETE.	Offers a rich set of data structures (strings, lists, sets, hashes, sorted sets).
Lacks built-in persistence mechanisms.	Provides persistence options (snapshotting, append-only file, replication).
Does not support complex queries or secondary indexes.	Supports more advanced operations like sorting, ranking, and filtering.
Focuses on high performance and low latency.	Balances performance with additional functionality and data structures.
No built-in way of replication, requires additional component.	Come with out-of-the-box replication using cluster, and as feature with **Sentinel**.

Consensus

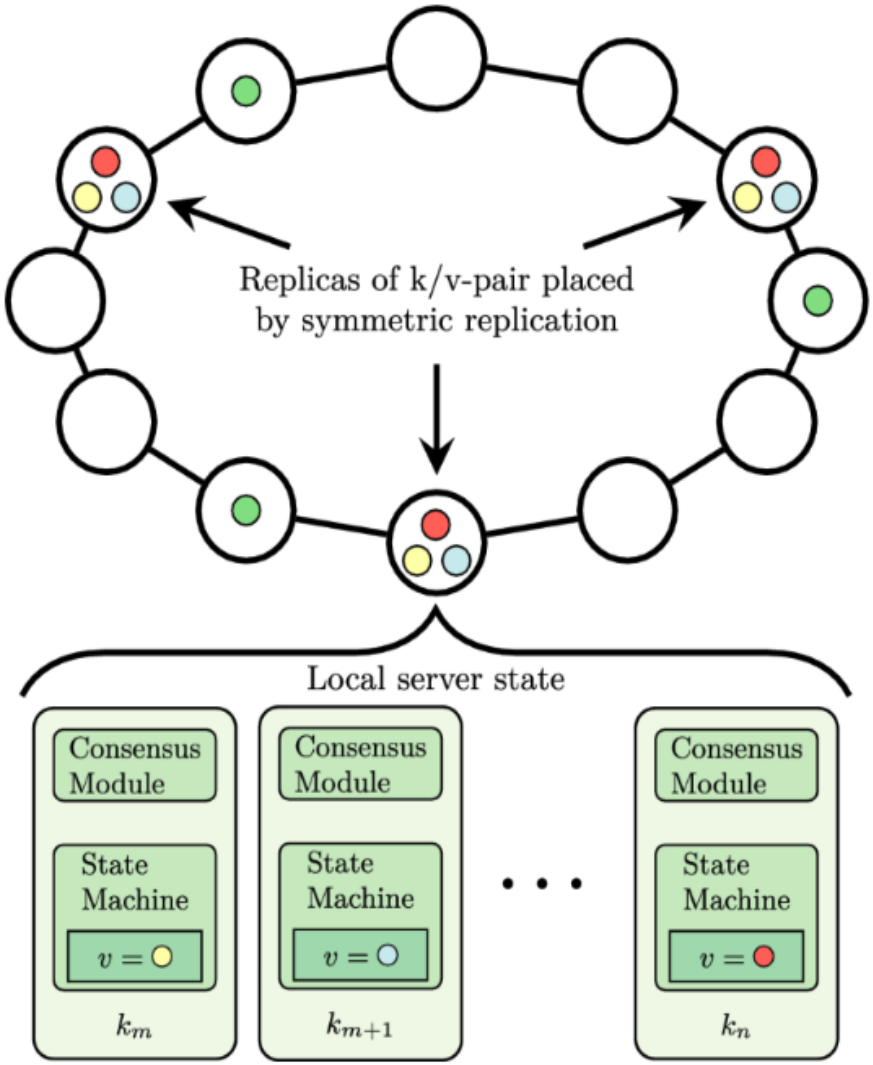
Replicated State Machine

Implementing a fault-tolerant service by replicating servers and coordinating client interactions with server replicas



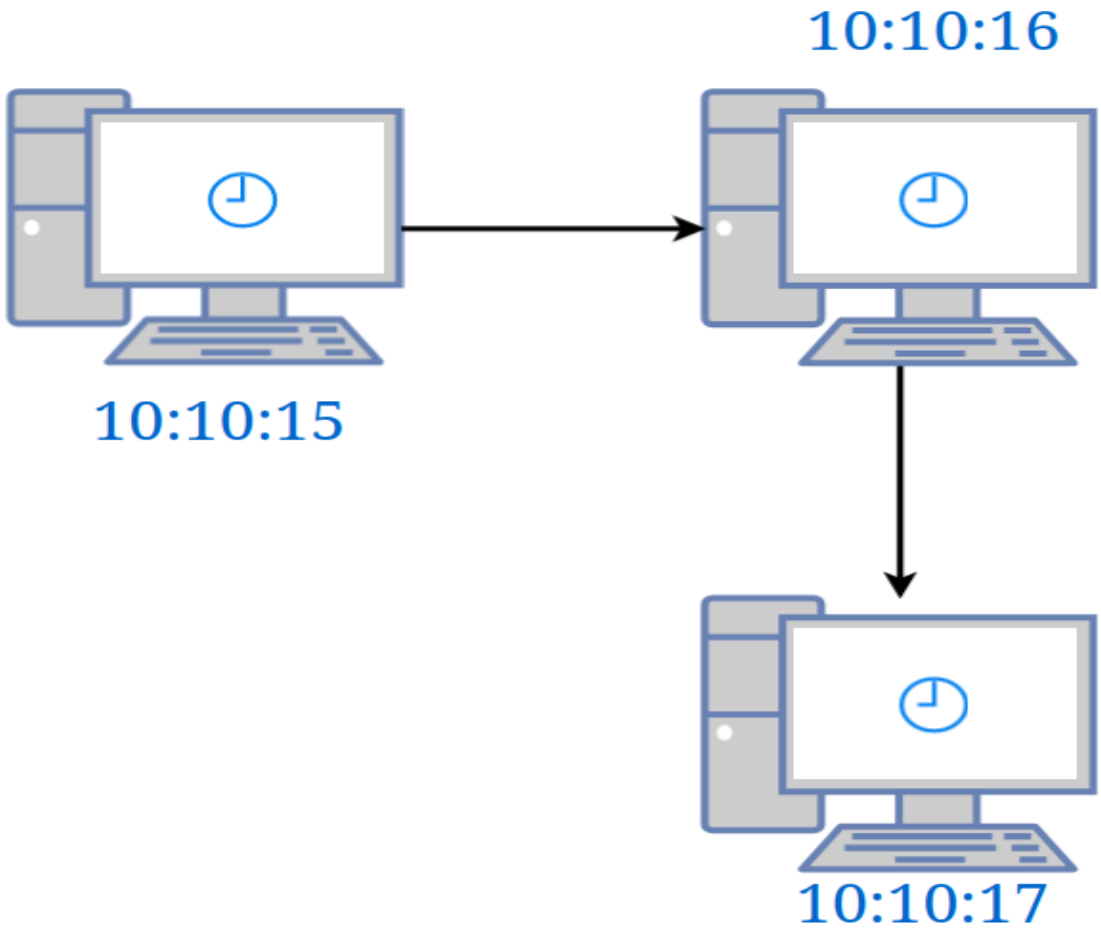
Algorithms

- **Paxos**
- **Raft**



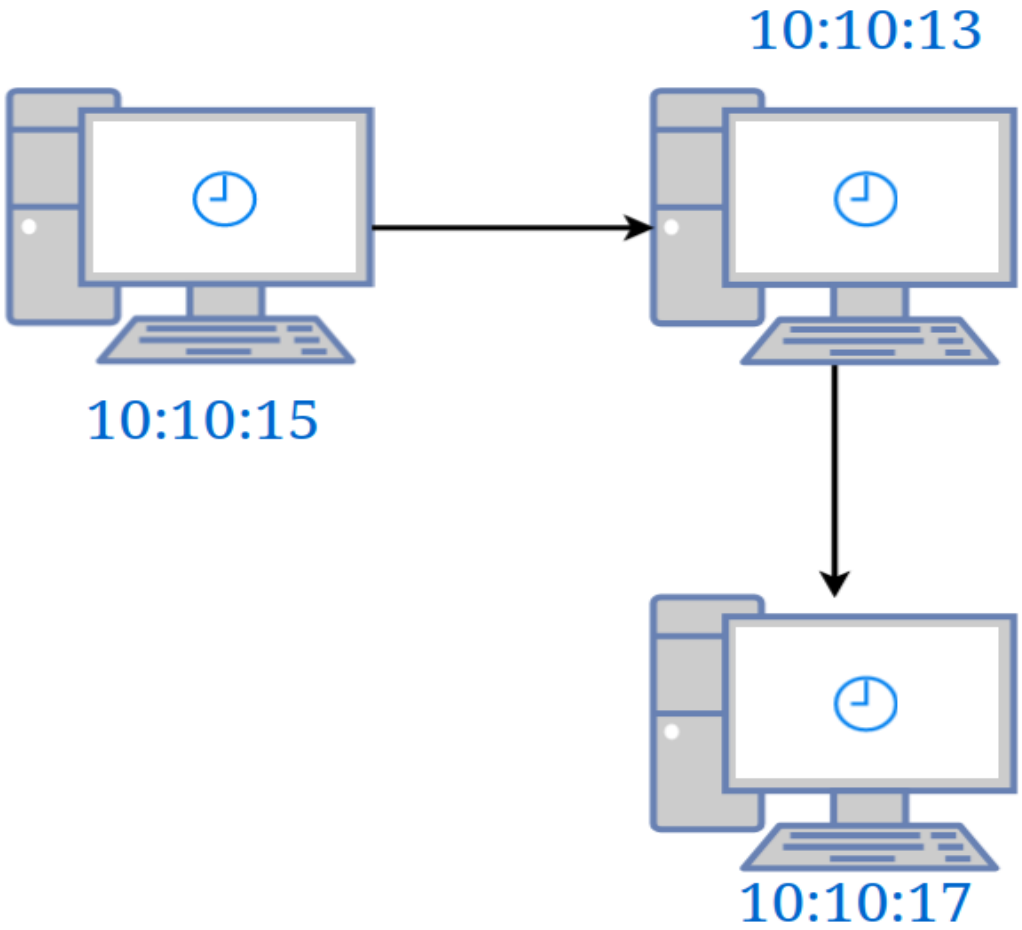
Natural/Physical Clock

What is the order of the events?



Natural/Physical Clock

What is the order of the events?



Natural/Physical Clock

Quartz Clock

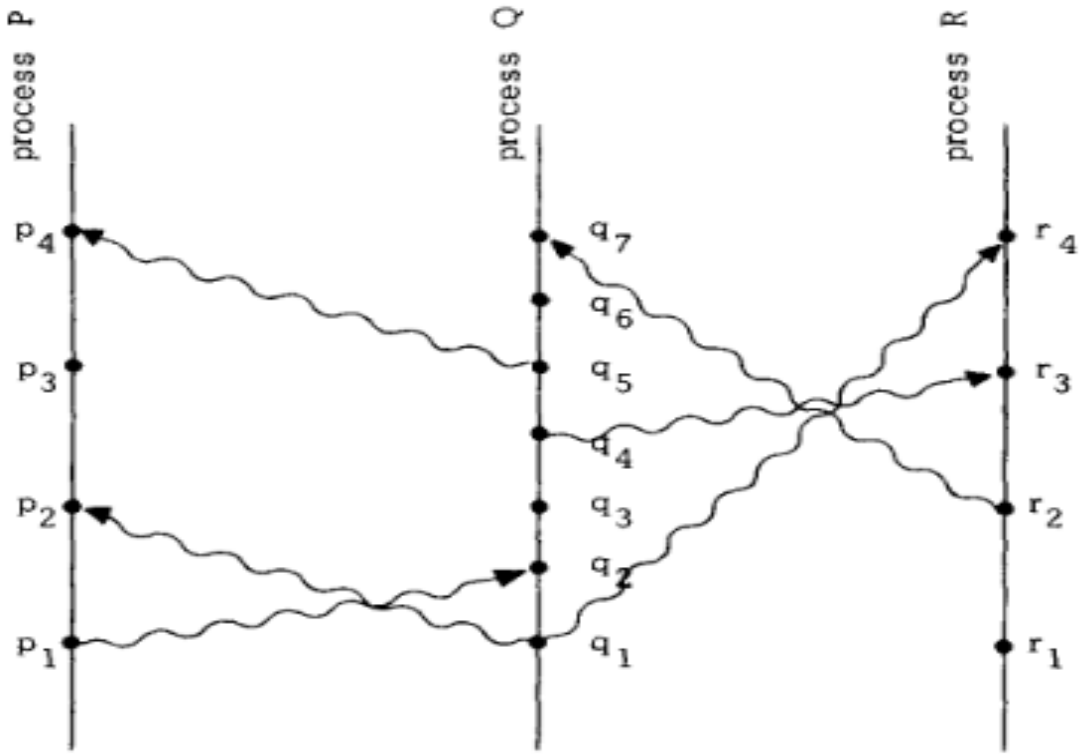
Logical Clock (Lamport Algorithm)

- Provide **happened before** relationship

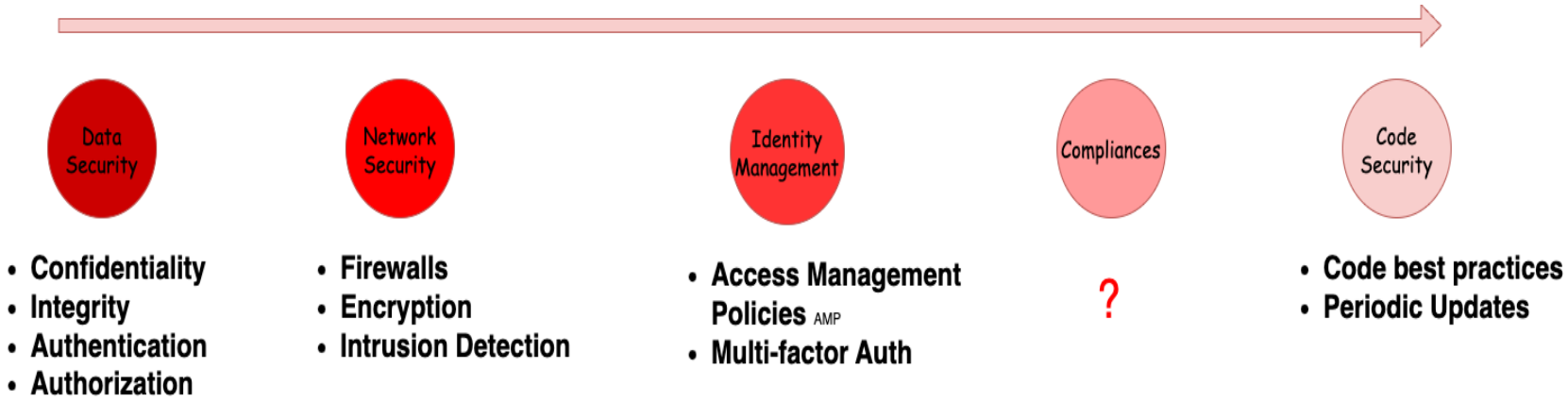
$$a \rightarrow b$$

- Total Ordering

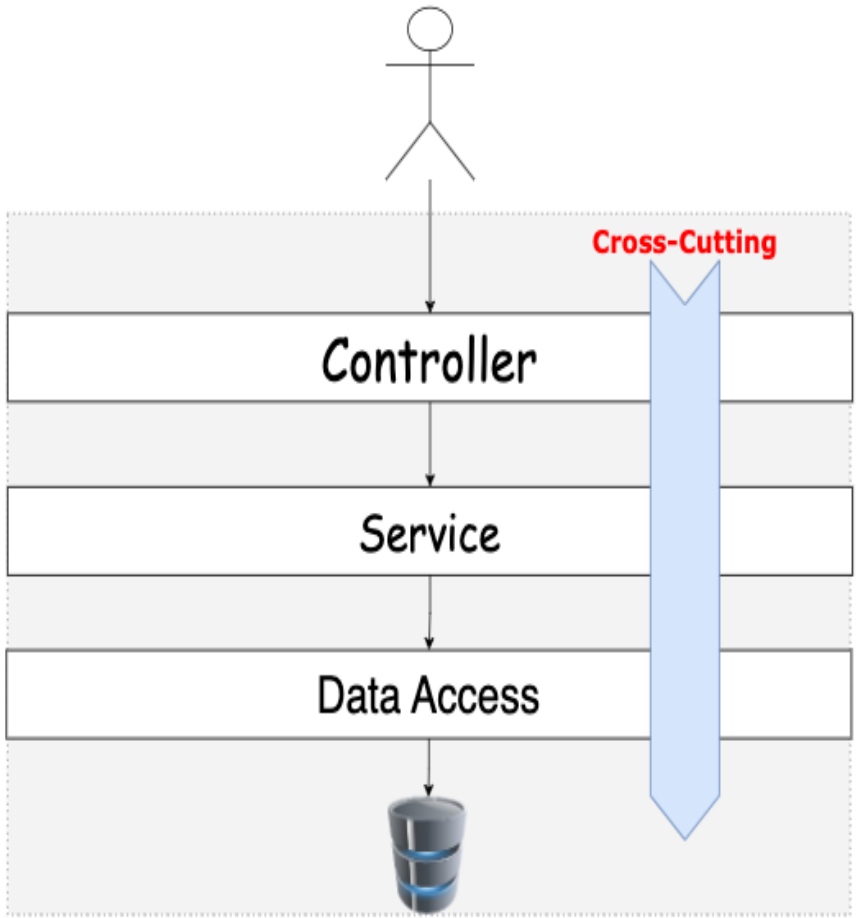
$$a \Rightarrow b \Longleftrightarrow a \rightarrow b$$



Distributed System Security



Distributed System Security



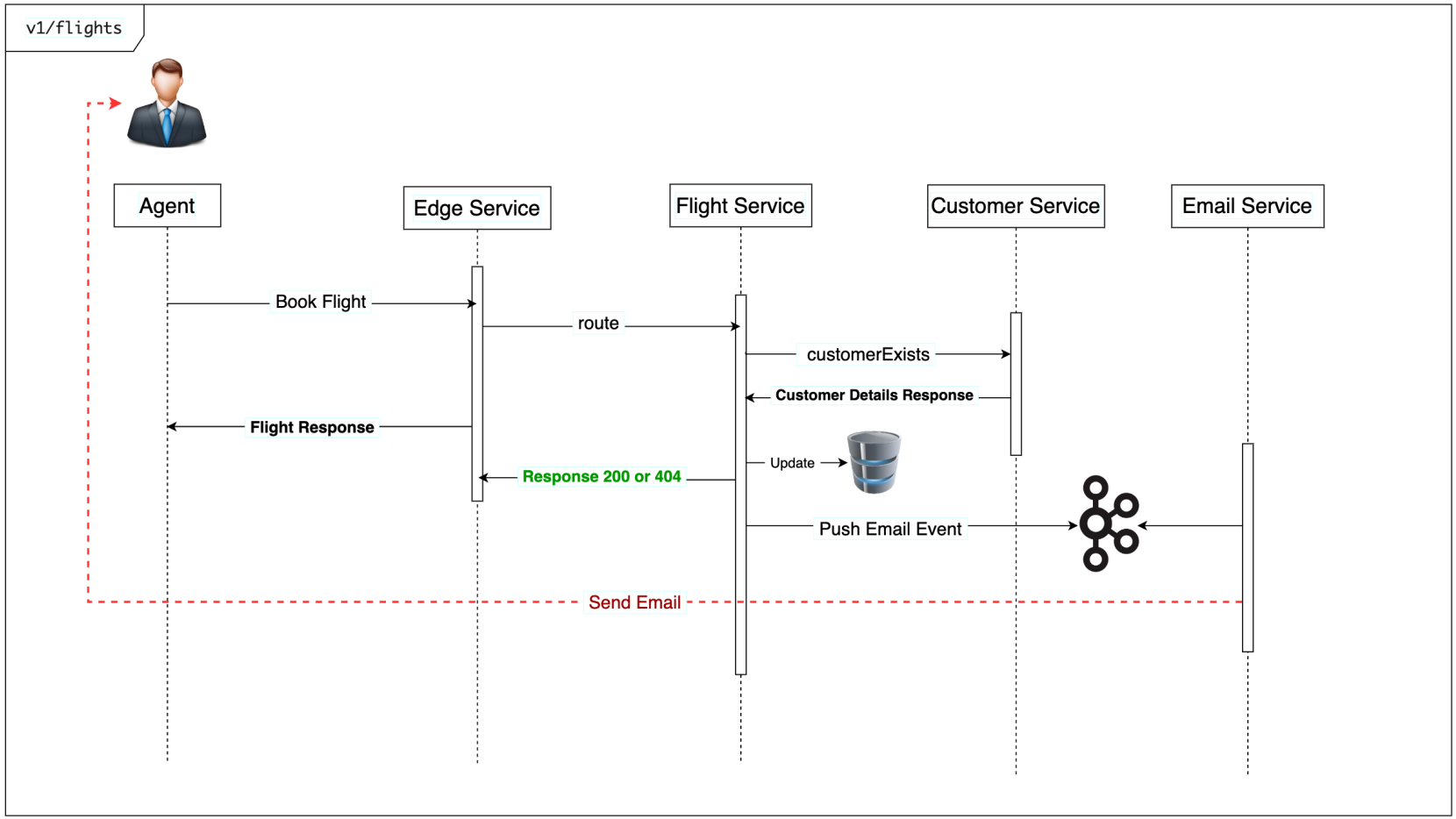
Cross-Cutting Concerns

- Logging
- Security
- Error Handling

Distributed System Monitoring

- Log aggregation and distributed tracing
- Metrics collection and monitoring tools (e.g., Prometheus, Grafana)
- Anomaly detection and performance optimization

API Documentations



Conclusion

- Recap of distributed systems concepts
- Overview of various distributed systems topics
- Further resources for exploring distributed systems in depth